DECODING MULTIPLE HMM SETS USING A SINGLE SENTENCE GRAMMAR

Field of Invention

This invention relates to speech recognition and more particularly to a speech recognition search method.

Background of Invention

Speech recognition devices are typically deployed in different acoustic environments. An acoustic environment refers to a stationary condition in which the speech is produced. For instance, speech signal can be produced by male speakers, female speakers, in office environment, in noisy environment.

A common way of dealing with multiple environment speech recognition is to train a set of Hidden Markov Models (HMM) for each environment. For example there would be a pronunciation set or network of HMMs (grammars)for male speakers and a set of HMMs for female speakers because the sounds or models for a male speaker are different from a female speaker. At the recognition phase, HMMs of all environments are decoded and the recognition result of the environment giving the maximum likelihood is considered as final results. Such a practice is very efficient in recognition performance. For example, if male/female separate models are not used, with the same amount of HMM parameters, the Word Error Rate (WER) will typically increase 70%.

More specifically, for a given sentence grammar, the speech recognizer is required to decode M (the number of environments) sets of HMMs each of which models a specific acoustic environment. In order to perform acoustic matching with each of the environments, recognition search methods typically (which include state-of-the-art recognizers as HTK 2.0) require a network of M sub-networks, as illustrated in Figure 1. Requiring M-sets of sentence network makes the recognition device more costly and requires much more memory.

Summary of Invention

A new speech recognition search method is described here, which needs only 1 out of the M subnetwork (sentence network) and yet gives the same recognition performance, thus reducing memory requirement for network storage by M-1/M. The speech recognition method includes a basic speaker independent grammar or network (sentence structure) and virtual symbols representing the network of expanded sets HMM sets where the pronunciation of each symbol is specified by a set of HMM states. The new recognizer builds recognition paths defined on the expanded symbols, and accesses the network using base-symbols, through proper conversion function that gives the base-symbol of any expanded symbols, and vice versa.

Description of Drawing:

In the drawing:

Figure 1 illustrates conventional recognizers require large network to recognize multiple HMM set;

Figure 2 illustrates a block diagram of the system according to one embodiment of the present invention;and

Figure 3 illustrates the main program loop.

Description of Preferred Embodiment

In the present application we refer a node in the network describing the sentence grammar as a *symbol.* For typical recognizers, a symbol has to be duplicated M-times in the network when M-sets of HMM are used. This is illustrated in Figure1, where three sets of sentence networks are depicted.

In accordance with the present invention a network is constructed to represent a merged version of the M networks that is speaker independent. For the male and female case this would be a merged version of the male and female networks and be gender-independent. The models for children may also be merged. Other environments may also be merged. We need to further decode specific HMMs such as those for male, female and child and combine with the generic (speaker independent) network where for both the male, female and child have the same nodes and transitions.

In applicant's method of decoding M HMM sets, two types of symbols are distinguished:

- *Base-symbols (α):* Symbol representing the basic grammar or network (i.e., the network before duplication for M-sets HMM). They have physical memory space for storage. This is generic (speaker independent) representing the nodes and transitions.
- *Expanded-symbols ($\tilde{a}$):* Symbols representing the network of M-1 expanded HMM sets. Their existence in the grammar network is conceptual. This symbol may represents for example the two sets of HMMs for male and female.

For each base-symbol in the network, there are M-1 corresponding expanded-symbols associated. The new recognizer builds recognition paths defined on the expanded-symbols, and accesses the network using base-symbols, through proper conversion function that gives the base-symbol of any expanded symbols.

Referring to Figure 2 there is illustrated the system according to one embodiment of the present invention. For the male and female combined case the generic network represented by the base symbol α is stored in memory 21. This provides the network structure itself. Also stored in memory 23 is a set of HMMs for male and a set for female for example. A set of HMMs may also be for child. The base symbol contains the sentence structure. The process is to identify the HMM to be used. For every incoming speech frame a main loop program performs a recognition path construction and update-observation-probability. The main loop program (see Figure 3) includes a path–propagation program 25 and an update-observation-probability program 27.

The function MAIN-LOOP program illustrated in Figure 3 performs recognition path construction for every incoming speech frame:

```
MAIN-LOOP (networks, models):
Begin
    For t = 1 to N Do
    Begin
        PATH-PROPAGATION (network, models, t):
        UPDATE-OBSERVATION-PROB (network, models, t);
    End
End
```

A path consists of a sequence of symbols, and the pronunciation of each symbol is specified by a set of hidden Markov model states. Consequently, a path can be either within-model-path or cross-model-path, which the decoding procedure constructs for each symbol:

```
PATH-PROPAGATION (network, hmms, t):
Begin
    For each active ã at frame t – 1 Do
    Begin
        (Δhmm, Δsym, ∀) = get-offsets (ã, network);
        hmm = hmms[hmm-code(symbol-list(network)[ ∀]) + Δhmm,];
        WITHIN-MODEL-PATH (hmm, . p_{t-1}^{ã}, p_t^{ã} );
```

CROSS-MODEL-PATH (hmms, network, $\tilde{a}$, $\forall$, Δ_{hmm}, Δ_{sym}, t, score ($p_t^{\tilde{a}}{}_{-1}$, EXIT-STATE));
End
End

where:

- p_t^s denotes the storage of path information for the expanded-symbol s at frame t.
- "get-offsets" gives the offset of HMM (Δ_{hmm}), offset of symbol (Δ_{sym}) and the base-symbol ($\forall$), given $\tilde{a}$ and a network.
- "symbol-list" returns the list of symbols of a network.
- "hmm-code" gives index of an hmm, associated to a symbol.
- Score (p, i) gives the score at state i of the symbol storage p. We keep what is the symbol and frame from which we are from t to t-1 and trace the sequence of the word . The nodes are constructed based on the model .

In the search algorithm for each frame time interval 1 to N for frame time t looks back at time t-1 and calculates to find out the base symbol. See Figure 2. From this to access the generic network 21 given the expanded symbol $\tilde{a}$ to get the offset of HMM (ΔHMM). Once the ΔHMM is determined, the HMM memory 23 can be accessed such that the HMM that corresponds to the male base or female is provided. Once the HMM is obtained the sequence of states within model path is determined and then the cross model path. The sequence of HMM states is constructed in the recognition path construction 25 in both the within HMM path and the between models. There are therefore two key functions for decoding, within-model-path construction and cross-model-path construction :

WITHIN-MODEL-PATH (hmm, p_{t-1}, p_t);
Begin
For each HMM state i of hmm Do
Begin
For each HMM state j of hmm Do
Begin
score (p_t, j) = score (p_{t-1}, i) + $a_{i,j}$;
from-frame (p_t, j) = from-frame (p_{t-1}, i)
from-symbol (p_t, j) = from-symbol (p_{t-1}, i)
End
End
End

where:

- $\forall_{i,j}$ is the transition probability from state i to state j.

When we do the within HMM path ,we need to do the storage of t and t-1. That sentence with the highest score is determined based on the highest transition log probability. This is done for every state in the HMM . (For each state j in the equation below. Once we arrive at the end we go back and find out what is the sequence of the symbols that has been recognized. This is stored.

CROSS-MODEL-PATH (hmms, network, $\tilde{a}$, $\forall \Delta_{hmm}$, Δ_{sym}, t, $*i$);
Begin
    For each next symbol s of $\forall$Do
    Begin
        hmm = hmms [hmm-code(symbol-list(network)[s]) + Δ_{hmm}];
        For each HMM initial state j of hmm Do
        Begin
            score ($p^{\square}$, j) = $*i$ X π (j);
            from-frame ($p^{\square}$, j) = $t - 1$;
            from-symbol ($p^{\square}$, j) = $\tilde{a}$;
        End
    End
End

For the cross model path we need for the next symbol s of α we need to consider all possible next symbols s . This is the true symbol s (knowledge of grammar that tells which symbol follows which symbol). We determine it's initial state or first HMM and we perform the sequence of HMM states for between states and add the transition probability (log probability)from one state to another. We use the π symbol for outside the states. We go back to the beginning and determine what is the symbol and frame from which we are from so that at the end we can go back and check the sequence of words. By doing this within and between we have constructed all the nodes.

Finally, once a path is expanded according to the grammar network, its acoustic score is evaluated:

    UPDATE-OBSERVATION-PROB (network, models, t);
    Begin
        For each active $\tilde{a}$ at frame t Do

Begin
(Δ_{hmm}, $\forall$) = get-true-symbol ($\tilde{a}$, network);
hmm = hmms[hmm-code(symbol-list(network)[$\forall$]) + Δ_{hmm}];
For each HMM state j of hmm Do
Begin
Evaluate score ($p_t^{\tilde{a}}$, j);
End
calculate score for $\tilde{a}$;
End
End
where:

- "get-true-symbol" returns the base-symbol of a expanded symbol.

These are all based on the model. The next step is to look at the speech to validate by comparison with the actual speech. This is done in the update-observation-probability program 27. See Figure 2. We need to find the HMM and for every HMM state we need to evaluate the score against the storage area at the time for the symbol α. The highest score is used. The best score models are provided.

RESULTS

This new method has been very effective at reducing the memory size. Below represents the generic grammar for 1-7 digit strings:

$digit = (zero | oh | one | two | three | four | five | six | seven | eight | nine)[sil];
$DIG = $digit [$digit [$digit [$digit [$digit [$digit [$digit]]]]]];
$SENT = [sil] $DIG [sil];

It says for we recognize zero or oh or one, or two etc. .It also says a digit is composed of a single digit,two digits etc.. It also says a sentence is on two etc. digits.

The grammar for the 1-7 digit strings for the old gender dependent way follows:

$digit_m = (zero_m | oh_m | one_m | two_m | three_m | four_m | five_m | six_m | seven_m | eight_m | nine_m)[sil_m];
$DIG = $digit_m [$digit_m [$digit_m [$digit_m [$digit_m [$digit_m [$digit_m]]]]]];
$SENT_m = [sil_m] $DIG_m [sil_m];
$digit_f = (zero_f | oh_f | one_f | two_f | three_f | four_f | five_f | six_f | seven_f | eight_f | nine_f)[sil_f];
$DIG_f = $digit_f [$digit_f [$digit_f [$digit_f [$digit_f [$digit_f [$digit_f]]]]]];
$SENT_f = [sil_f] $DIG_f [sil_f];

$S = $SENT_m | $SENT_f;

This is twice the size of the generic grammar.

The purpose is to calibrate resource requirement and verify that the recognition scores are bit-exact with multiple grammar decoder. Tests are based on ten files, 5 male 5 female.

For the grammars above , respectively, a single network grammar of sentence and a multiple (two, one for male, one for female) network grammar of sentence.

COMPUTATION REQUIREMENT

Due to the conversion between base and expanded symbols, the search method is certainly more complex than the one requiring M-set of networks. To determine how much more computation is needed for the sentence network memory saving, the CPU cycles of top 20 functions are counted, and In are shown in Table 1(excluding three file I/O functions). It can be seen that the cycle:

Item	multiple-grammar	Single-grammar
allocate_back_cell	1603752	1603752
coloring_beam	1225323	1225323
coloring_pending_states *	2263190	2560475
compact_beam_cells	2390449	2390449
cross_model_path *	2669081	2847944
fetch_back_cell	10396389	10396389
find_beam_index	7880086	7880086
get_back_cell	735328	735328
init_beam_list	700060	700060
logGaussPdf_decode	19930695	19930695
log_gauss_mixture	2695988	2695988
mark_cells	13794636	13794636
next_cell	898603	898603
path_propagation *	1470878	1949576
Setconst	822822	822822
update_obs_prob *	5231532	5513276
within_model_path	3406688	3406688

Table 1 CPU cycle comparison for top time-consuming functions (UltraSPARC-II). '*' is introduced to indicate an item that has a changed value.

Consumption for most functions stays the same. Only four functions showed slight changes. Table 2 summarizes cycle consumption and memory usage. The 1.58% is spent on calculating the set-index, and can be further reduced by storing the indices. However, the percent increase is so law that at this time it might not be worth-doing to investigate the other alternative – CPU efficient implementation.

Item	multiple-grammar	single-grammar	increase
TOP CYCLES	78115500	79352090	1.58
NETWORK SIZE	11728	5853	-50.0

Table 2 Comparison of multiple-grammar vs. single-grammar (memory-efficient implementation).